Name: Limpin, Jholim Jats R.

Section: NW-301

1. Localvariables

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Localvariables {
    uint public i;
    bool public b;
    address public myAddress;

    function foo() external {    infinite gas
        uint x = 123;
        bool f = false;

        x += 456;
        f = true;

        i = 123;
        b = true;
        myAddress = address (1);
    }
}
```

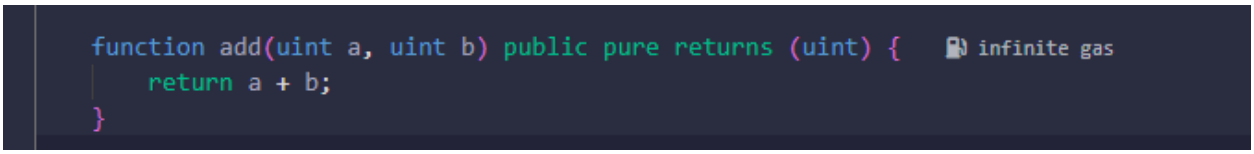
2. GlobalVarialbes

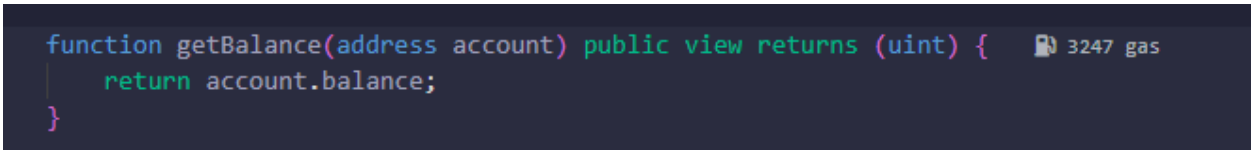
```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract GlobalVarialbes {
    function globalVars() external view returns (address, uint, uint) {    infinite gas
        address sender = msg.sender;
        uint timestamp = block.timestamp;
        uint blockNum = block.number;
        return (sender, timestamp, blockNum);
    }
}
```

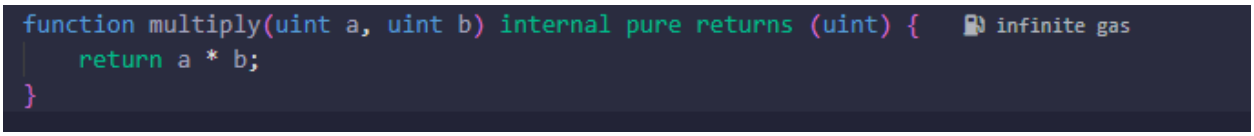
Reflection:

The contracts show the difference between global and local variables in Solidity. Global variables like msg.sender, block.timestamp, and block.number connect the contract to the blockchain environment, giving context such as caller identity, time, and block height. Local variables, like x and f inside a function, exist only temporarily, while state variables such as i, b, and myAddress persist on-chain. Together, they

teach that Solidity balances blockchain context, short-term logic, and permanent storage to build reliable applications.







Reflection :

Pure functions are isolated and predictable, serving as logical tools that rely only on inputs, while view functions act as observers, allowing contracts to read and reflect the current state of the blockchain. Together, they show how Solidity separates computation from state awareness, teaching that clarity in function type ensures both efficiency and trust in smart contract design.